AD-A242 046

(2)

Designing a Workstation-Based Conferencing System Using the Real-Time Producer/Consumer Paradigm

TR90-040

October, 1990

DTIC
ELECTE
OCT 24 1991
S D
D

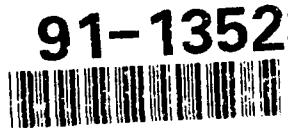
Kevin Jeffay, F. Donelson Smith

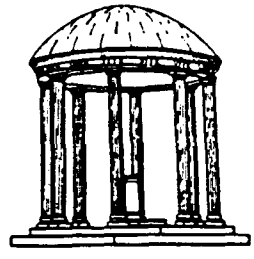
N00014-86-K-0680

ONR

This document has been approved for public release and sale; its distribution is unlimited.

91-13523





The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

UNC is an Equal Opportunity/Affirmative Action Institution.

91 1018 003

Designing a Workstation-Based Conferencing System Using the Real-Time Producer/Consumer Paradigm

Kevin Jeffay F. Donelson Smith

Department of Computer Science
University of N. Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

August 1990

COPY INSPECTED 4

1. Introduction

The Computer Science department of the University of North Carolina at Chapel Hill is involved in a federally funded initiative to study the use of computer technology to support collaboration among scientists and professionals [Smith et al. 90]. Our project is interdisciplinary; involving computer scientists, cognitive psychologists, and an anthropologist.

One component of this project is the construction of a distributed, real-time, video conferencing system on a network of high-performance workstations. This note concerns the technological issues involved in realizing a workstation-based conferencing system. The emphasis of our research is on the development of operating system and network support for real-time communication of digital audio and video streams.

From a technological standpoint, the goal is to support multiple, concurrent streams of digital audio and video in a distributed network of computer workstations. These streams may either form disjoint conferences within the network or involve one node (workstation) in multiple simultaneous conferences. Our approach is to view the problem as one of real-time resource allocation and control. With respect to real-time performance, our system should be indistinguishable from the more traditional analog (i.e., non-computer based) realization of a conferencing system. This will be assessed not only by feedback from users but also through analytical modeling. In the latter case we plan to demonstrate by rigorous techniques that the system adheres, in all cases, to a set of well defined

STATEMENT A PER TELECON
RALPH WACHTER ONR/CODE 1133
ARLINGTON, VA 22217
NWW 10/23/91

A-1

performance parameters. For example, one parameter will be a specification of worst case response time for a packet of compressed audio/video data delivered end-to-end.

To construct and analyze such a system we have adopted a real-time system design discipline based on a paradigm of process interaction called the *real-time producer/consumer (RTP/C) paradigm* [Jeffay 89a]. The RTP/C paradigm defines a semantics of inter-process communication that provides a framework for reasoning about the real-time behavior of programs. This semantics is realized through an application of some recent results in the theory of deterministic scheduling and resource allocation.

This report is organized in the following sections. Section 2 describes the broader context of this work and motivates our use of computer workstations for a desk-top conferencing system. Section 3 describes the RTP/C paradigm in greater detail and discusses its application to the problem of supporting digital audio and video in a distributed system. Section 4 outlines both the hardware and software environment of the physical system that we are building. Sections 5 and 6 conclude this report with brief discussions of our long range goals, expected contributions and current status.

2. Motivation

2.1. Collaboration and Conferencing

The driving problem that sustains our research program is computer and communication support for collaboration by groups of scientific or technical professionals [Lederberg & Uncapher 89, Smith, et al. 90, Calingaert, et al. 90]. This research is one part of the broader field of computer-supported cooperative work (CSCW)[ACM 86, ACM 88, ACM 90, Greif 88].

Why do we need collaborations and how can computer and communication support help? No single individual has all of the expertise, the information, the diversity of point of view, or the time to carry out large-scale projects. Instead, we must synthesize these resources in a group. Frequently, however, the right people are not all in the same location. Physical separations, even relatively small ones, can seriously inhibit interactions among people, *e.g.*, among departments scattered across a large university campus, across a multi-site corporation, or even within a multi-story building. Facilities for effective communication and sharing of information are, we believe, the essential ingredients in a successful collaboration-support system.

Our view is that people working together usually create some concrete, often complex, conceptual artifact that is expressed in words, drawings, images, etc. These artifacts take many forms: a book manuscript, a computer program, a proposal describing planned scientific experiments, a patient's medical records, a manual of policy and procedures, or a user's manual for a computer. Our research is based on the premise that much of collaboration is concerned with creating a complex conceptual artifact, reaching a shared understanding of it, and agreeing on changes to it as it evolves over time.

One critical issue, then, is how to provide effective access to shared, evolving materials to an ever-changing group of participants, most of whom may work semi-independently and may be separated geographically. We refer to this notion of largely autonomous activity by members of a group as "asynchronous" collaboration. We believe that a shared hypermedia database [Brandes & Lewerentz 85, Campbell & Goodman 87] is the most promising tool for storing and accessing the artifacts created by a group of collaborators — it is the foundation of our support for asynchronous collaboration. Users of our system will be able to create, browse, and link (in the hypermedia sense) elements of many types of information including text, drawings (2D graphics), and recorded sight and sound.

Whereas collaborators may spend a great deal of time working independently, many activities require periods of direct person-to-person interaction — exploring, questioning, proposing, reviewing, negotiating, agreeing. We refer to such episodes of direct interactions (conferences) among members of a group as "synchronous" collaboration. Synchronous collaborations provide the fine-grained transfer of information within the context of a long-term asynchronous collaboration. These conferences will be most productive if all participants simultaneously have full access to their shared computer-stored materials. Furthermore, it should be convenient to begin and sustain conferences without elaborate scheduling or, better yet, without leaving one's office. Consequently, we believe that shared visual workspaces implemented on graphics-based workstations [Abdel-Wahab, et al. 88, Ensor, et al. 88, Lantz 86] and augmented by voice and video communications are the most promising technologies for supporting synchronous collaboration. While we don't expect network communication to replace all face-to-face meetings, better support systems should help make distributed collaboration an effective and efficient choice for a broad spectrum of creative tasks.

Shared visual workspaces are abstractions that denote a collection of objects (*e.g.*, documents, images, programs) and the tools used to view and change them. Each participant in a collaboration can logically share the same view of these objects (and the

same tools for operating on the objects). A shared visual workspace facility permits multiple users to work together with the same object(s) at the same time. It supports a form of distributed electronic meeting in which team members work at their individual workstations, linked by the network. In this context, they all receive the same visual information. While one is performing operations in the shared workspace, the others see those interactions. Cooperation is managed through various conventions, such as "passing the chalk," to keep participants from interfering with one another. The shared visual workspaces must be supplemented by audio (and possibly video) media for communication so participants can approximate face-to-face discussions of their thinking and their actions.

While not yet a complete solution for multi-user applications or conferencing systems, the X Window System [Scheifler & Gettys 86] offers many desirable properties. The most appealing aspect is the ability to interconnect distributed applications with user interfaces over networks. It is also widely available, its components interoperate in many heterogeneous environments, and it can be adapted to support multi-user collaborations. Our facilities for synchronous collaboration (including sight and sound) are designed for integration with the X Window System producing a system that provides workstation-based multi-media conferencing. Our research emphasizes an evaluation of the effectiveness and added benefit of rich, but expensive, facilities such as audio and full-motion video. We can do this by configuring systems that support shared workspaces only, shared workspaces plus voice, and shared workspaces plus voice and video, and then observing differences in behavior for groups using the different systems.

2.2. Why digital media?

All the requirements for media in collaboration systems could be met using a combination of conventional digital and analog (audio/video) technology. For example, one could equip a workstation with a LAN adapter for X Window System protocols and database access, an adapter for analog video (e.g., the Parallax card), and an audio adapter. In buildings already wired for data, voice, and CATV communications (as three separate technologies), these facilities could be used to extend conferencing to office workstations. In fact, our building, Sitterson Hall, has these wiring types (and more) installed; we plan to use them for early prototypes of a collaboration system. Why, then, are we interested in digital media? One important reason is the cost advantage of a single technology that can support communications for both people and computers — a single "information receptacle" in the office wall into which one can plug a workstation and (with the required cameras,

microphones, and digitizing adapters) turn it into a multi-media computing and conferencing tool.

Perhaps a more compelling reason is that, if audio and video are represented in digital form, the data can be handled in new ways. Software can be written to implement functions that now require specialized hardware in teleconferencing systems (voice-activated controls to put the current speaker's image in a window; multi-image windows "quad-split" so that up to four participants are simultaneously visible). With all-digital media, a software conferencing system can be designed so each user can dynamically customize communications (e.g., whose image is displayed, size and arrangement of windows, update rates, etc.). Other interesting functions are also possible such as the ability to "cut" a live segment of video or audio from a conference window and "paste" it into a node of the hypermedia database. Of course we can also have the ability to access digitally recorded audio and video from the hypermedia data storage and transmit them in real time over the network for play-back.

2.3. System design for digital media

Digital sight and sound media have two properties that present challenging system design problems — bandwidth consumption and continuous, periodic demands. Recent advances in compression technology have brought bandwidth demands for digital video within the capabilities of advanced workstations. For example, Intel's Digital Video Interactive (DVI) [Ripley 89] system can compress color, full-motion (30 frames per second) video in real time with VCR-like quality. The resulting bandwidth requirement for transmitting the compressed stream over a LAN is approximately 1.1 Mbps (storing a two minute sequence of compressed video requires approximately 17 Mbytes). In a workstation used for collaboration support, many competing demands for bandwidth (e.g., from the window system or file access) must be satisfied concurrently with those from one or more streams of video and/or audio.

Digital audio and video are often called "continuous media" because the information is continuously changed, usually at regular intervals. Intuitively, we can think of these media as a continuous stream of new "data packets" that arrive periodically, each replacing the one before it. For example, a new video image must be captured, compressed (producing with DVI technology about 4.5 Kbytes), transmitted, decompressed, and displayed every 33 milliseconds in order to give the remote viewer a natural perception of motion. This means that the temporal properties of audio and video streams are part of the semantic content and

should be preserved for maximal fidelity. Some relaxation of this requirement is possible, however, because people's perceptions have a small degree of tolerance — a rare discontinuity will usually not be noticed. Because a person is present at both ends of a continuous-media stream in a conferencing application, it is important to closely approximate the "look and feel" of a normal conversation. This means that both end-to-end delays and irregularities in the streams must be carefully controlled. Further, because the participants are likely to be discussing actions in their shared visual workspaces, both the graphics windows and the continuous-media windows must be updated concurrently. These requirements naturally lead to use of real-time scheduling in operating systems for our workstations.

We will limit our system to supporting groups working within a distributed computing environment of workstations equipped for multiple digital media (audio, video, graphics) which communicate using a single logical local-area network (includes multiple LANs interconnected by MAC-layer bridges). Examples of LAN systems that appear well suited to the conferencing application are the 16-Mbps token ring and the 100-Mbps FDDI ring. In addition to bandwidth, two particularly desirable functions of these networks are priority service and group addressing. Our research also anticipates wide-area networks with Gigabit speeds that will make continuous media widely distributable. The problems of managing continuous media in these networks are being addressed as part of the DASH project [Anderson, et al. 89] We anticipate using those results in a future extension of our system.

3. Programming Model

In this section we describe the underlying formal model upon which we intend to base the construction of our initial system. The physical operating environment is assumed to be a simple, generic distributed computing system consisting of a set of workstations connected by a single logical local area network as described above. The workstations are assumed to be homogeneous in the sense that each workstation supports a common set of programming abstractions implemented according to a set of well-defined parameters. The basis of the program abstractions and their implementation is a paradigm of real-time system design and analysis called the *real-time producer/consumer (RTP/C) paradigm* [Jeffay 89a]. The remainder of this section provides an overview of the RTP/C paradigm and discusses its contributions to the problems of integrating digital audio and video in a distributed system.

3.1. The Real-Time Producer/Consumer Paradigm

The real-time producer/consumer paradigm defines a semantics of process interaction that enables one to reason about the real-time behavior of a program. A *program* is viewed as a directed graph where vertices represent processes and edges represent unidirectional communication channels. Processes exchange messages along communication channels. Processes may be either sequential programs that execute on a single processor or physical processes in the environment external to the processor that communicate with internal processes via interrupts.

Each channel in a graph defines a producer/consumer relationship between two processes. Each channel is characterized by a *rate* indicating the worst case minimum inter-arrival time that can be observed between any two messages sent on the channel. The RTP/C paradigm stipulates that over a well-defined interval, a consumer of messages on a channel must consume messages at precisely the rate at which they are produced. Logically, a message sent (produced) on a channel must be received (consumed) *before the next message is sent.* Conceptually, a producer defines a discrete time domain for a channel. The emission of messages on this channel corresponds to the "ticks" of a discrete time clock. If a pair of interconnected processes adheres to the RTP/C paradigm, then, relative to the channel's discrete time clock, a producer cannot tell the difference between a consumer that is infinitely fast and one that simply obeys the RTP/C paradigm. In this manner the RTP/C paradigm allows one to specify one form of real-time computation; ignoring implementation details such as processor speeds.

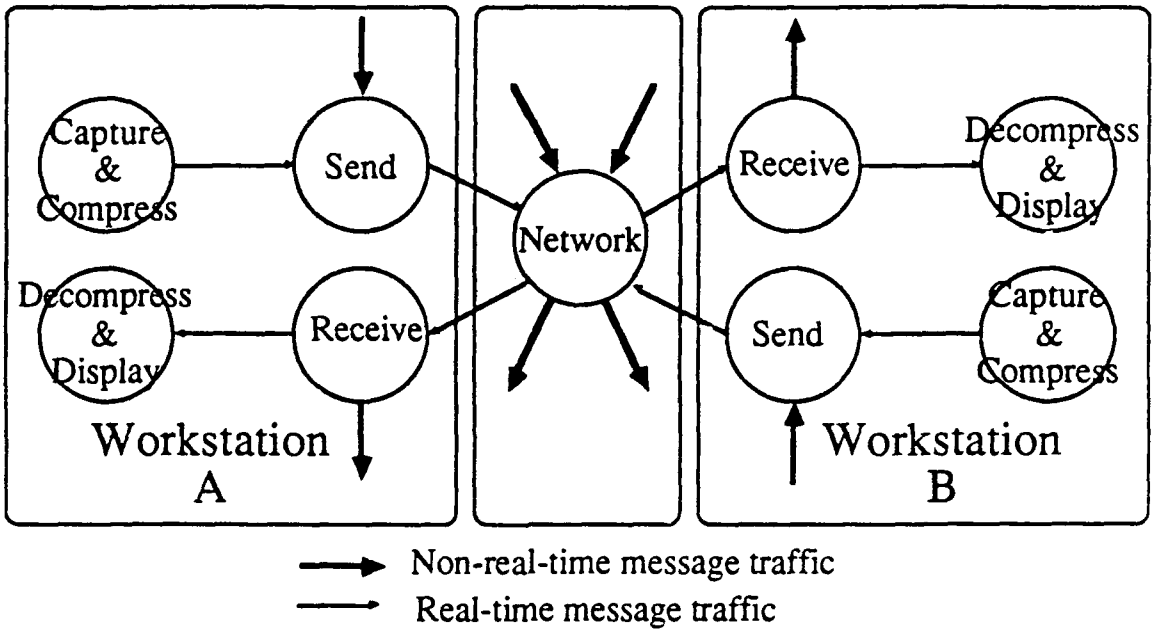
Given a specification of the processing resources available in the system, the key challenge is to implement a process graph such that all interconnected pairs of processes are guaranteed to adhere to the RTP/C paradigm. We have studied the theoretical foundations of this problem extensively [Jeffay 89b, Jeffay et al. 90, Jeffay 90]. A process graph is modeled as a set of tasks that make repetitive requests for execution and have a deadline for the completion of each execution request. Moreover, tasks may have constraints on their execution due to the presence of critical sections. With respect to the formal model, we have developed an optimal algorithm for sequencing such tasks on a single processor. The algorithm is optimal in the sense that it can schedule a set of tasks in such a manner that all execution requests of all tasks will complete execution before their respective deadlines whenever it is possible to do so. This work has resulted in a decision procedure for determining whether a given set of processes can be executed such that all interconnected

pairs of processes are guaranteed to adhere to the RTP/C paradigm. The parameters to this decision procedure are the maximum amounts of computation time required to consume a message on each channel and the worst case inter-arrival times of messages on channels. The former class of parameters are a function of the algorithms used within a process and the speed of the processor. The inter-arrival times are ultimately a function of the rates at which data (messages) enter the system. A methodology for computing these parameters is presented in [Jeffay 89a]. These results are applied to the problem of designing a distributed real-time system by mapping appropriate subgraphs of a process graph onto physical processors or the network.

The RTP/C paradigm provides a framework both for expressing processor-time-dependent computations and for reasoning about the real-time behavior of programs. For example, for an arbitrary path through a graph one can compute bounds on the time required for a message, or sequence of messages, to propagate between the source and sink processes. Other traditional performance metrics such as throughput are implicit in the model. The adoption of the RTP/C paradigm as a programming abstraction for real-time systems also allows one to assess design tradeoffs such as the introduction of a faster processor, or the effect of restructuring a computation.

3.2. Supporting Digital Audio and Video with the RTP/C Paradigm

The programming model we have described can be viewed as a type of data-flow model. As such it is well suited to applications whose real-time constraints arise from the need to process continuous streams of data that enter the system. For a workstation-based conferencing application, the entire system, including software processes, I/O devices, and the interconnection network, is represented as a directed process graph. For example, in the high-level design shown below, the compression unit of a DVI card on workstation *A* might appear as the source vertex in the graph and the decompression unit on a DVI card on workstation *B* might appear as a sink vertex in the graph. A path between these nodes might traverse processes representing a user application process on workstation *A*, a network interface process, the physical network process, a network interface process on workstation *B*, a user application process on workstation *B*, and finally a DVI control process. Each process may be an abstraction of a collection of (lower-level) processes (e.g. a subgraph). In addition, processes may consume message streams from non-real-time processes (those that do not require the RTP/C paradigm for correct operation).



Given a real-time system design, it is important to assess and understand its expected real-time performance (assuming some underlying implementation strategy). For example, for the process graph shown above, one might wish to determine bounds on the latency for a compressed video frame (or a sequence of frames) to traverse a path from a compression source to a decompression sink. The RTP/C paradigm provides a framework for performing such analyses given information on the rates at which messages are sent, and the cost (in terms of execution time) of processing individual messages. The analysis reflects the fact that distributed applications such as ours naturally lend themselves to hierarchical decomposition. We assume a system design includes a static mapping of process subgraphs to processors and I/O devices. The analysis of a program's real-time behavior proceeds by separately considering process subgraphs that physically execute on the same processor. For each subgraph that corresponds to a schedulable device, response time bounds are computed for paths of interest through the subgraph. These results are combined hierarchically to determine actual end-to-end latency and response time bounds for the overall system.

The adoption of the RTP/C paradigm as a programming abstraction for systems that manipulate digital audio and video will allow us to rigorously analyze the real-time behavior of the system we construct and to asses tradeoffs in design choices such as the introduction of a faster processor, or the effect of restructuring a computation by joining a set of

processes into a single process or by decomposing a process into a set of processes. Moreover we will be able to perform calculations such as deriving limits on the number of real-time streams of continuous media that our system (collection of hardware and software resources) can handle simultaneously.

3.3. Research Issues

Our application of the RTP/C paradigm to a distributed network of workstations motivates the consideration of a number technical problems in the field of distributed real-time computing. The primary research issues we see our work addressing are:

- integration of hard and soft real-time processing constraints, and
- loosely-coupled processor and resource allocation problems.

The RTP/C paradigm, and indeed much of the real-time scheduling literature, assumes a boolean measure of temporal correctness for a real-time system. To be correct, a system must ensure that all instances of all timing constraints are respected by its implementation. (Such systems are frequently referred to as *hard-real-time systems*.) Such a correctness metric is clearly not required for all real-time applications. For example, within the context of a digital audio and video application, there are well defined sampling and replay rates that are necessary and sufficient for a faithful reproduction of the analog equivalent. It is clear, however, that "short" and "infrequent" periods of "small" deviations from these rates are tolerable. The problem is to both specify and make effective use of the degradation in real-time performance that can be tolerated by an application. We would like to be able to provide a framework for expressing best and worst case performance scenarios and to integrate this information into a resource allocation policy so that in times of insufficient processing resources, intelligent choices can be made concerning the degradation of the system. For example, in times of heavy load, instead of reducing the rate at which data is processed, for some applications it might make sense to maintain a specified output rate but degrade (reduce or eliminate) the processing of individual outputs.

The trade-offs between real-time response requirements and acceptable degradation of data integrity point to a deeper problem: the integration of hard and soft-real-time constraints within a system. The problem here is to allocate resources in such a manner that the "hard-real-time" constraints are guaranteed to be adhered to without sacrificing the implicit performance requirements of the soft-real-time tasks. For example, a common implementation paradigm in real-time systems is the "foreground/background" structuring

of real-time tasks. In these systems hard-real-time tasks execute with highest priority. Processor cycles unused by hard-real-time tasks are time-sliced among soft-real-time tasks. While this provides acceptable response times to the hard-real-time tasks, the soft-real-time tasks often exhibit unacceptable performance. (In fact it is often because of this that users recognize the implicit real-time nature of tasks previously considered to be "non-real-time.") One possible avenue to exploit is the fact that hard-real-time tasks often require only a bounded response time. When such a task makes a request for execution it must be completed before some well defined deadline. In particular, there is no value to the task in completing as early as possible within this interval. This suggests a means to improve the response times of soft-real-time tasks through the use of a "just-in-time" scheduler. The design and analysis of such a scheduler represents an open problem.

The problem of accommodating hard and soft real-time processing constraints is independent of the underlying system architecture. The second problem our research is addressing concerns the allocation of resources in an environment with multiple schedulable processors, such as a distributed system. In such a system the problem is to effectively allocate resources so as to satisfy processing constraints that span processor boundaries. For example, in a workstation-based conferencing system, an important constraint will be to ensure a particular level of throughput and bounded latency for packets of compressed audio and video that are generated on one machine and displayed on a second machine. As these packets will be manipulated by devices such as the compression hardware, CPU, and network controller on the originating machine, and the network controller, CPU, and decompression hardware at the destination machine, the responsibility for ensuring the throughput and latency constraints are met will be distributed throughout the system. The goal is to operate these devices in concert so as to realize the desired overall performance. Real-time scheduling and resource allocation on multiple processors is a notoriously hard problem [Dertouzos & Mok 89]. Indeed, many simple problems are known to be intractable [Garey & Johnson 77]. Nevertheless, we must develop an effective strategy for allocating processing resources in this environment.

While initially motivated by a particular approach to a particular problem (the use of the RTP/C paradigm and the desire to support continuous media in a distributed system), the research problems we have identified are central to the field of distributed real-time computing and as such will be applicable to a wide range of systems beyond our conferencing application.

4. Proposed Implementation and Experiments

Our prototype implementation environment uses IBM PS/2 machines (Intel 80386 processors) with Intel ActionMedia 750 capture and delivery adapters (DVI technology). Each workstation is also equipped with a video camera and microphone for input to the DVI capture adapter, along with a VGA graphics display and speakers for output from the DVI delivery adapter. The PS/2 machines are interconnected by a 16-Mbit token ring. This configuration allows us to conduct experiments involving multiple concurrent streams of digital continuous media along with more conventional types of data (*e.g.*, remote file access). The multiple concurrent streams of continuous media are required to support multiple simultaneous conferences (current generation DVI cards may not be able to handle multiple conferences on a single workstation; future versions should provide more capability). The base operating system for these machines is IBM PC-DOS.

At the lower levels of the system we will be constructing operating systems support for the RTP/C paradigm as real-time task-scheduling extensions to the base PC-DOS. This involves the implementation of new kernel functions extending PC-DOS with a tasking *model, scheduling algorithms, and resource* allocation policies. The specific kernel functions we are constructing have two novel features:

- use detailed knowledge about the applications executed, such as process graph topology, to provide minimal response times for message propagation, and
- exploit the properties of the scheduling policies we employ to implement tasks without the usual overhead of maintaining separate run-time stacks for each task.

A prototype of these kernel functions has been constructed [Jeffay & Poirier 90].

5. Contributions

Ultimately we see our efforts contributing to the design and analysis of distributed real-time systems. Specific contributions envisioned include:

- a system for achieving guaranteed response times in a distributed system,
- a framework for the integration and accommodation of hard and soft-real-time processing requirements,

- a framework for the a priori analysis of the real-time behavior of systems, and
- an operating systems kernel that demonstrates the efficacy of the processor and resource allocation policies we have developed.

6. Summary and status

We are currently installing and testing the hardware for the prototype configuration described above. A model and analysis for continuous media support based on the real-time producer/consumer model is being developed concurrently with an implementation of a run-time system (a real-time scheduler extension to PC-DOS) that supports the analysis used in the model.

7. References

[Abdel-Wahab et al. 88]
Abdel-Wahab, H M., Guan, S.-U., Nievergelt, J., *Shared Workspaces for Group Collaboration: An Experiment using Internet and UNIX Interprocess Communications*, IEEE Communications, Vol. 26, No. 11, (1988), pp. 10-16.

[Anderson et al. 89]
Anderson, D.P., Tzou, S.-Y., Wahbe, R., Govindan, R., Andrews, M., *Support for Continuous Media in the DASH System*, University of California at Berkeley, Computer Science Division (EECS), Report No. UCB/CSD 89/537, October 1989.

[ACM 86] *Proceedings of the Conference on Computer-Supported Cooperative Work*, ACM, Austin, Texas, December 1986.

[ACM 88] *Proceedings of the Second Conference on Computer-Supported Cooperative Work*, ACM, Portland, Oregon, September 1988.

[ACM 1990] *Proceedings of the Third Conference on Computer-Supported Cooperative Work*, ACM, Los Angeles, CA, October 1990.

[Brandes & Lewerentz 85]
Brandes, T., Lewerentz, C., *GRAS: A Non-standard Data Base System within a Software Development Environment*, Proceedings GTE Workshop on Software Engineering Environments for Programming-in-the-Large, June 1985, pp. 113-121.

[Brooks 87] Brooks, F.P., Jr., *No Silver Bullet -- Essence and Accidents of Software Engineering*, In Information Processing 86, Elsevier, Amsterdam (1986), pp. 1069-1076. Reprinted in Computer, Vol. 20, No. 4, (April 1987), pp. 10-19.

[Calingaert et al. 90]
Calingaert, P., Abdel-Wahab, H.M., Bollella, G., Guan, S.-U., Smith, F.D., Smith, J.B., *Synchronous and Asynchronous Collaboration: A Program of Research*, University of North Carolina at Chapel Hill, Department of Computer Science, Technical Report TR90-024, May 1990.

[Campbell & Goodman.87]
Campbell, B., Goodman, J.M., *HAM: A General-purpose Hypertext Abstract Machine*, Proceedings Hypertext '87, November 1987, pp. 21-32.

[Dertouzos & Mok 89]
Dertouzos, M.L., Mok, A.K.-L., *Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks*, IEEE Trans. on Soft. Eng., Vol. SE-15, No. 12, (December 1989), pp. 1497-1506.

[Ensor et al. 88]
Ensor, J.R., Ahuja, S.R., Horn, D.N., Lucco, S.E., *The Rapport Multimedia Conferencing System - a Software Overview*, Proceeding of the Second IEEE Conference on Computer Workstations, Santa Clara, CA, March, 1988.

[Garey & Johnson 77]
Garey, M.R., Johnson, D.S., *Two-Processor Scheduling with Start-Times and Deadlines*, SIAM J. Computing, Vol. 6, No. 3, (September 1977), pp. 416-426.

[Greif 88] Greif, I., Computer-Supported Cooperative Work: A Book of Readings, Morgan Kaufmann Publishing Co., Palo Alto, CA, 1988.

[Jeffay 89a] Jeffay, K., *The Real-Time Producer/Consumer Paradigm: Towards Verifiable Real-Time Computations*, Ph.D. Thesis, University of Washington, Department of Computer Science, Technical Report #89-09-15, September 1989.

[Jeffay 89b] Jeffay, K., *Analysis of a Synchronization and Scheduling Discipline for Real-Time Tasks with Preemption Constraints*, Proc. Tenth IEEE Real-Time Systems Symp., Santa Monica, CA, December 1989, pp. 295-305.

[Jeffay 90] Jeffay, K., *Scheduling Sporadic Tasks with Shared Resources in Hard-Real-Time Systems*, University of N. Carolina at Chapel Hill, Department of Computer Science, Technical Report TR90-038, August 1990. (Submitted for publication.)

[Jeffay et al. 90]
Jeffay, K., Anderson, R., Martel, C.U., *On Optimal, Non-Preemptive Scheduling of Periodic and Sporadic Tasks*, University of N. Carolina at Chapel Hill, Department of Computer Science, Technical Report TR90-019, March 1990. (Submitted for publication.)

[Jeffay & Poirier 90]
Jeffay, K., Poirier, D., *An Implementation and Application of the Real-Time Producer/Consumer Paradigm*, University of N. Carolina at Chapel Hill, Department of Computer Science, Technical Report TR90-039, August 1990.

[Lantz 86] Lantz, K.A., *An Experiment in Integrated Multimedia Conferencing*, Proceedings, Conference on Computer-Supported Cooperative Work, Austin, TX, December 1986, pp. 267-275.

[Lederberg & Uncapher 89]
Lederberg, J., Uncapher, K., (eds.), *Towards a National Collaboratory: Report of an Invitational Workshop at the Rockefeller University*, March 17-18, 1989. Distributed by the National Science Foundation.

[Ripley 89] Ripley, G.D., *DVI - A Digital Multimedia Technology*, CACM, Vol. 32, No. 7, (July 1989), pp. 811-822.

[Scheifler & Gettys 86]
Scheifler, R.W., Gettys, J., *The X Window System*, ACM Transactions on Graphics, Vol. 5, No. 2, (April 1986), pp. 79-109.

[Smith et al. 90]
Smith, J.B., Smith, F.D., Calingaert, P., Jeffay, K., Holland, D., Hayes, J.R., *Building and Using a Collaboratory: A Foundation for Supporting and Studying Group Collaborations*, National Science Foundation Grant No. IRI 90-15443, April 1990.